Markov Text Generator

Yuanjing Zhu

September 26, 2022

1 Introduction

In mtg.py, I wrote a class object with two methods: whatisnextword and finish_sentence. THe first function will iterate the whole corpus and store the combination of n tokens and its frequency. Then the finish_sentence function will append the next most likely word given the provided tokens until it matches ".", "?", "!" or the total length of the tokens exceeds 10.

2 Some examples

Here are some test examples for this bare-bone Markov text generator using Jane Austen's Sense and Sensibility as corpus.

2.1 Sample test

In this test case, the input tokens are ["she", "was", "not"], using 3-gram model and deterministic is True. The generated sentence is ["she", "was", "not", "in", "the", "world", "."], which is the same as the given answer.

```
words = finish_sentence(
        ["she", "was", "not"],
        3,
        corpus,
        deterministic=True)
print(words)

Output: ['she', 'was', 'not', 'in', 'the', 'world', '.']
```

2.2 Deterministic and stochastic modes

When deterministic is False, the text generator will pick the next word randomly from the appropriate distribution. Therefore, with the same input sentence ["i", "would"] and using a 3-gram model, the generated sentences in the two cases are different.

```
F_det1 = finish_sentence(
        ["i", "would"],
        3,
        corpus,
        deterministic=False)
print(F_det1)

F_det2 = finish_sentence(
        ["i", "would"],
        3,
        corpus,
        deterministic=False)
print(F_det2)
```

Generated texts are as follows:

n	Quantity
Input	[”i”, ”would”]
Output1	[’i’, ’would’, ’not’, ’give’, ’offence’, ’.’]
Output2	[’i’, ’would’, ’not’, ’be’, ’deceived’, ’as’, ’to’, ’norland’, ’half’, ’its’]

2.3 Different n

Next, I used different seed words to test the impact of n on this text generator. The input tokens are [’i’, ’do’, ’not’] and the deterministic is True. For the 5 cases, n increases from 2 to 6 and the output sentences are shown in the following table.

n	Output
2	[’i’, ’do’, ’not’, ’be’, ’a’, ’very’, ’well’, ’,’, ’and’, ’the’]
3	[’i’, ’do’, ’not’, ’know’, ’what’, ’you’, ’are’, ’very’, ’much’, ’to’]
4	[’i’, ’do’, ’not’, ’know’, ’what’, ’you’, ’and’, ’mr.’, ’willoughby’, ’got’]
5	[’i’, ’do’, ’not’, ’know’, ’what’, ’you’, ’and’, ’mr.’, ’willoughby’, ’will’]

When n is small (n= 2, 3), the generated sentence doesn’t make any sense. This is because the model generated the next word only based on the previous 1/2 words. As n increases, the sentence grammar is getting correct and it starts to become a meaningful sentence.